**Assign same task to multiple threads**

class MyThread extends Thread{

public void run(){

System.out.println("task1...");

}

}

class Main{

public static void main(String args[]){

MyThread t1=new MyThread();

MyThread t2=new MyThread();

MyThread t3=new MyThread();

t1.start();

t2.start();

t3.start();

}

}

**Assign different task to multiple threads**

class MyThread1 extends Thread{

public void run(){

System.out.println("task1...");

}

}

class MyThread2 extends Thread{

public void run(){

System.out.println("task2...");

}

}

class Main{

public static void main(String args[]){

MyThread1 t1=new MyThread1();

MyThread2 t2=new MyThread2();

t1.start();

t2.start();

}

}

**Deadlock**

When a thread holds a resource and waits for another resource to be released by second thread, the second thread holding a resource and waiting for a resource to be released by first thread, then in such case both the thread will be waiting and they never execute. This is called deadlock.

class Test{

public synchronized void show1(Best b){

System.out.println("Thraed1 start execution of show1()..");

try{

Thread.sleep(6000);

}

catch(Exception e){

}

System.out.println("Thraed1 is trying to call display method of Best class");

b.display();

}

public synchronized void display(){

System.out.println("display() method of Best class");

}

}

class Best{

public synchronized void show2(Test t){

System.out.println("Thraed2 start execution of show2()..");

try{

Thread.sleep(6000);

}

catch(Exception e){

}

System.out.println("Thraed2 is trying to call display method of Test class");

t.display();

}

public synchronized void display(){

System.out.println("display() method of Test class");

}

}

class Deadlock extends Thread{

Test t = new Test();

Best b = new Best();

public void m1(){

this.start();

t.show1(b);

}

public void run(){

b.show2(t);

}

public static void main(String args[]){

Deadlock d = new Deadlock();

d.m1();

}

}

**ThreadGroup**

class MyThread implements Runnable{

public void run() {

System.out.println(Thread.currentThread().getName());

}

}

class ThreadGroupDemo{

public static void main(String[] args) {

MyThread runnable = new MyThread();

ThreadGroup tg1 = new ThreadGroup("Parent ThreadGroup");

Thread t1 = new Thread(tg1, runnable,"one");

t1.start();

Thread t2 = new Thread(tg1, runnable,"two");

t2.start();

Thread t3 = new Thread(tg1, runnable,"three");

t3.start();

System.out.println("Thread Group Name: "+tg1.getName());

tg1.list();

}

}

**Calling Private Method Using Refelection API**

import java.lang.reflect.Method;

class Student{

private int getAge(){

return 10;

}

}

class Main{

public static void main(String args[])throws Exception{

Student t = new Student();

Method privateMethod = Student.class.getDeclaredMethod("getAge");

privateMethod.setAccessible(true);

int age = (int)privateMethod.invoke(t);

System.out.println("Age of Student: " + age);

}

}

**Difference between new and newInstance()**

**new Operator**

class Student{

Student(){

System.out.println("constructor...");

}

}

class Main{

public static void main(String args[]){

Student s = new Student();

}

}

**newInstance()**

class Student{

Student(){

System.out.println("constructor...");

}

}

class Main{

public static void main(String args[])throws Exception{

Class c = Class.forName("Student");

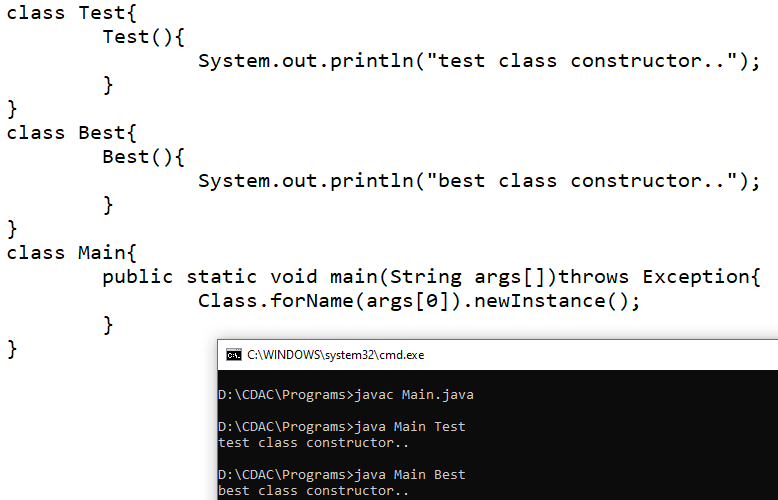
c.newInstance();

}

}

**Note**

If class name is given at runtime then we have to use newInstance() method to crate object.



**Data Class**

**import** java.util.Calendar;

**import** java.util.Date;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Date dt = **new** Date();

System.***out***.println(dt);

Calendar c = Calendar.*getInstance*();

System.***out***.println(c.get(Calendar.***DATE***)+"-"+(c.get(Calendar.***MONTH***)+1)+"-"+c.get(Calendar.***YEAR***));

System.***out***.println(c.get(Calendar.***HOUR***)+":"+c.get(Calendar.***MINUTE***)+":"+c.get(Calendar.***SECOND***));

}

}

**Simple Date Format**

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Date dt = **new** Date();

SimpleDateFormat sdf = **new** SimpleDateFormat("dd/MM/YYYY");

String dt2 = sdf.format(dt);

SimpleDateFormat sdf2 = **new** SimpleDateFormat("dd-MMM-YYYY hh:mm:ss");

String dt3 = sdf2.format(dt);

System.***out***.println(dt2);

System.***out***.println(dt3);

}

}